探索 Iris（鸢尾花）数据集

CS245　数据科学基础　陆朝俊

叶泽林 515030910468

1　问题描述

统计学主要研究事物的数量方面，目的是探索数据集的数量特征。而统计学中的描述统计学借助图表或概括性的数值将数据集展示为清晰可理解的形式。在之前的研究中，已经使用了图表对 Adults 数据集进行了探索和可视化展示，这次研究的主要目标是探索 Iris 数据集，并通过一些概括性的数值对其进行展示，详细目标如下：

1. 探索 Iris 数据集的基本属性（如数据集总体描述、数据维数、特征名称等）；
2. 探索各特征的最小值、最大值、均值、中位数、标准差；
3. 探索各特征之间，以及特征与目标之间的相关性（相关系数）。

2　解决方案

2.1　数据集及其基本属性的获取

2.2　各特征的分析

2.3　相关性分析

3　结果展示

3.1　Iris 数据集基本属性

表 1: Iris 数据集基本属性

属性	值
实例的数据个数	150
实例的数据类型	numpy.ndarray
实例的数据维数	(150, 4)
特征名	萼片长度，萼片宽度，花瓣长度，花瓣宽度 (单位均为 cm)
实例的类别值	0,1,2
实例的类别维数	(150,)
类别名称	setosa(清风藤), versicolor(云芝), virginica(锦葵)

3.2　各特征的数值描述

详细结果可参见表 2 (考虑到有些统计量无单位，因此在表格中不把单位显式地表示出，各特征单位均为 cm)。

表 2: Iris 数据集各特征的数值描述

	萼片长度	萼片宽度	花瓣长度	花瓣宽度
最小值	4.3	2.0	1.0	0.1
最大值	7.9	4.4	6.9	2.5
均值	5.843	3.054	3.759	1.120
中位数	5.80	3.00	4.35	1.30
标准差	0.825	0.432	1.759	0.761
方差	0.681	0.187	3.092	0.579
极差	3.6	2.4	5.9	2.4
下四分位数	5.1	2.8	1.6	0.3
上四分位数	6.4	3.3	5.1	1.8

3.3 各特征及特征与目标之间的相关性

表 3: Iris 数据集各特征及特征与目标之间的相关性

	萼片长度	萼片宽度	花瓣长度	花瓣宽度
萼片长度	1.000	-0.109	0.872	0.818
萼片宽度	-0.109	1.000	-0.421	0.357
花瓣长度	0.872	-0.421	1.000	0.963
花瓣宽度	0.818	-0.357	0.963	1.000
目标	0.783	-0.420	0.949	0.956

A 附录

A.1 Iris 数据集的其余基本属性

A.1.1 Iris 数据集的基本描述

Iris 数据集基本描述

```
Notes
-----
Data Set Characteristics:
    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the
      class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
```

```
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica
    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20  0.76     0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

This is a copy of UCI ML iris datasets.
http://archive.ics.uci.edu/ml/datasets/Iris

The famous Iris database, first used by Sir R.A Fisher

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the
   field and
is referenced frequently to this day.  (See Duda & Hart, for example.)
    The
data set contains 3 classes of 50 instances each, where each class
   refers to a
type of iris plant.  One class is linearly separable from the other 2;
    the
latter are NOT linearly separable from each other.

References
----------
   - Fisher,R.A. "The use of multiple measurements in taxonomic
     problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "
        Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda,R.O., & Hart,P.E. (1973) Pattern Classification and Scene
     Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New
     System
     Structure and Classification Rule for Recognition in Partially
        Exposed
```

```
  Environments". IEEE Transactions on Pattern Analysis and Machine
  Intelligence, Vol. PAMI-2, No. 1, 67-71.
- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE
   Transactions
  on Information Theory, May 1972, 431-433.
- See also: 1988 MLC Proceedings, 54-64. Cheeseman et al"s
   AUTOCLASS II
  conceptual clustering system finds 3 classes in the data.
- Many, many more ...
```

A.2 主要代码

Iris 数据集探索代码

```python
from sklearn import datasets
import stats
import numpy as np
import xlwt

class IrisAnalyzer(object):
    def __init__(self):
        # iris data
        self.iris = datasets.load_iris()
        self.data = self.iris.data
        self.target = self.iris.target
        print(self.iris.DESCR)

        # save results as excel tables
        self.wbk = xlwt.Workbook()
        self.char_sheet = self.wbk.add_sheet('各特征分析')
        self.corr_sheet = self.wbk.add_sheet('相关分析')
        self.metrics = {'max': np.max, 'min': np.min, 'avg': np.mean,
                        'median': np.median, 'ptp': np.ptp,
                        'std': np.std, 'var': np.var, 'q1': stats.quantile,
                        'q3': stats.quantile}

        # initialize the excel tables
        for i, name in enumerate(self.iris.feature_names):
            self.char_sheet.write(0, i + 1, name)
        for i, m in enumerate(self.metrics.keys()):
            self.char_sheet.write(i + 1, 0, m)
        for i, name in enumerate(self.iris.feature_names):
            self.corr_sheet.write(0, i + 1, name)
        for i, m in enumerate(self.iris.feature_names):
            self.corr_sheet.write(i + 1, 0, m)
        self.corr_sheet.write(i + 2, 0, 'target')

        print('----------analyzer started----------')

```

```python
    def run(self):
        # starting to analyze Iris dataset
        self.basic_att()
        self.char_analysis()
        self.corr_analysis()

        self.wbk.save('results.xls')
        print('——————results saved——————')

    def basic_att(self):
        # get basic attributes of Iris dataset
        with open('basic_att.txt', 'w') as f:
            f.write(self.iris.DESCR + '\n\n')

            f.write('Type of data: ' + str(type(self.data)) + '\n\n')
            f.write('Shape of data: ' + str(self.data.shape) + '\n\n')
            f.write('Feature names: ' + str(self.iris.feature_names) + '\n\n')

            f.write('Target: ' + str(self.target) + '\n\n')
            f.write('Type of target: ' + str(type(self.target)) + '\n\n')
            f.write('Shape of target: ' + str(self.target.shape) + '\n\n')
            f.write('Target names: ' + str(self.iris.target_names) + '\n\n')

    def char_analysis(self):
        # analyzing each of characters
        for i in range(self.data.shape[1]):
            char_data = self.data[:,i]
            for j, m in enumerate(self.metrics.keys()):
                if not m in ['q1', 'q3']:
                    self.char_sheet.write(j + 1, i + 1, self.metrics[m](char_data))
                elif m == 'q1':
                    self.char_sheet.write(j + 1, i + 1, self.metrics[m](char_data, p=0.25))
                elif m == 'q3':
                    self.char_sheet.write(j + 1, i + 1, self.metrics[m](char_data, p=0.75))

    def corr_analysis(self):
        # analyzing the correlationships
        for i in range(self.data.shape[1]):
            char_data = self.data[:, i]

            # corr among characters
            for j in range(self.data.shape[1]):
                self.corr_sheet.write(j + 1, i + 1, np.corrcoef(
```

```
                   char_data, self.data[:, j])[0][1])

            # corr between characters and target
            self.corr_sheet.write(j + 2, i + 1, np.corrcoef(char_data,
                self.target)[0][1])

def debug():
    # just used to debug, ignore it
    pass

if __name__ == '__main__':
    ia = IrisAnalyzer()
    ia.run()
```